VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Webový 3D editor

Web-based 3D editor

2015 Miroslav Tomášek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Miroslav Tomášek**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Webový 3D Editor
Web-based 3D Editor

Zásady pro vypracování:

Cílem bakalářské práce je návrh a implementace webového 3D editoru pro zobrazení a editaci prostorové scény. Při implementaci 3D editoru budou využity technologie HTML5 (zejména WebSocket a XHR2), CSS3 a především WebGL (s použitím JavaScriptu). Editor bude mít následující vlastnosti:

1. Otevření a uložení scény ve formátu obj:
   a) Lokálního úložiště uživatele (v .zip formátu s přiloženými texturami,
   b) ze vzdáleného úložiště (s využitím WebSocket).
2. Editace viditelných objektů (polygon meshes):
   a) Automatické zobrazení polygonové síťoviny objektu,
   b) označení jednotlivých plošek, případně vertex bodů polygonové síťoviny,
   c) geometrická transformace plošek a vertex bodů,
   d) extruze označených plošek a vertex bodů,
   e) smazání vybraných plošek, příp. vertex bodů,
   f) nastavení složek materiálu (ambientní, difúzní, spekulární), změna textury.
3. Transformace:
   a) Objektů scény (posun, rotace, změna měřítka),
   b) kamery včetně trackball,
   c) změna pozice bodového osvětlení.
4. Přidání/odebrání bodového osvětlení.
5. Možnost vícenásobného zobrazení scény.

Seznam doporučené odborné literatury:

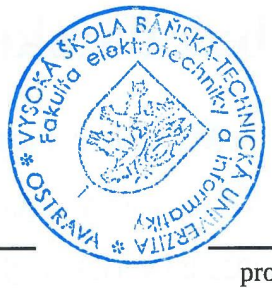
[1] Matsuda, Kouchi, and Rodger Lea. WebGL programming guide : interactive 3D graphics programming with WebGL. Upper Saddle River, NJ: Addison-Wesley, 2013
[2] Anyuru, Andreas. Professional WebGL programming developing 3D graphics for the web. Chichester, U.K: John Wiley & Sons, 2012
[3] Jamsa, K. (2014). Introduction to Web development using HTML 5. Burlington, MA: Jones & Bartlett Learning

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Lukáš Zaorálek**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

V Ostravě 7.5. 2015 ..... Tomášek .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7.5. 2015 ..... Tomášek .....

Rád bych zde poděkoval svému vedoucímu bakalářské práce Ing. Lukáši Zaorálkovi za odborné vedení a rady. Poděkování také patří mé partnerce a celé rodině za podporu.

Abstrakt

Tato bakalářská práce je zaměřena na implementaci webového 3D editoru, který zvládne zobrazit a editovat prostorové scény. Při implementaci jsou využity technologie HTML5, CSS3 a především WebGL s použitím JavaScriptu (framework BabylonJS). Editor poskytuje možnosti načítání a ukládání scény ve formátu obj. Lze využít vícenásobného zobrazení scény. Dále editor umožňuje editaci bodových světel, viditelných objektů a jejich transformaci.

Klíčová slova: HTML5, CSS3, WebGL, 3D editor, BabylonJS, vertex

Abstract

This Bachelor thesis is concentrate to implementation of web-based 3D editor, which can viewing and editing spatial scenes. Technologies of HTML5, CSS3 and especially WebGL using JavaScript (framework BabylonJS) are utilized for implementation. Editor provide options of loading and saving scenes in obj format. It is possible to show scene in multiviewing mode and editting light points or visible objects and their transformation.

Keywords: HTML5, CSS3, WebGL, 3D editor, BabylonJS, vertex

Seznam použitých zkratek a symbolů

3D	–	trojrozměrný
2D	–	dvourozměrný
HTML	–	HyperText Markup Language
HTML5	–	HyperText Markup Language 5
XHTML	–	eXtensible HyperText Markup Language
CSS3	–	Cascading Style Sheets 3
WebGL	–	Web-based Graphics Library
XHR2	–	XMLHttpRequest Level 2
WWW	–	World Wide Web
W3C	–	World Wide Web Consortium
CERN	–	Center Européenne pour la Recherche Nucléaire, Evropské centrum jaderného výzkumu
BRDF	–	Bidirectional Reflectance Distribution Function, Dvousměrná odrazová distribuční funkce

Obsah

Seznam obrázků

1 Úvod

Tématem bakalářské práce je vytvořit webovou aplikaci, která dokáže pracovat s objekty v 3D prostředí. Celý 3D editor je vytvořen za pomoci technologií HTML5, CSS3 a WebGL s použitím JavaScriptu. Rozhodl jsem se využít framework BabylonJS, který má již spoustu WebGL funkcí pro práci v 3D prostoru naimplementovanou.

Jak již bylo zmíněno, tak prostředí, ve kterém editor pracuje s objekty, je trojrozměrné. To znamená, že v tomto prostředí mají objekty svůj objem stejně jako objekty v reálném světě. Existují i jiná prostředí jako je třeba dvourozměrné, kde objekty mají obsah, ale nemají objem, např. čtverec, kruh, atd. Základem pochopení 3D prostoru je tzv. kartézská souřadná soustava. Tato soustava v prostoru využívá tří rozměrů, jimiž jsou délka, šířka a hloubka (či výška). Tyto rozměry jsou v prostoru konvenčně označovány jako osy x,y a z. Poloha v prostoru je označována souřadnicemi pomocí číselných hodnot, např. (1, -5, 3). Souřadnice vypovídají o umístění bodu, který je na ose x vzdálen od počátku o 1 jednotku, na ose y o -5 jednotek a na ose z o 3 jednotky od počátku. Počátek neboli střed prostoru má souřadnice (0, 0, 0). Negativní hodnoty vyjadřují opačný směr a tím umožňují využívat prostor od mínus nekonečna do nekonečna.

Základním stavebním kamenem 3D grafiky jsou body a trojúhelníky složené ze tří bodů, ze kterých můžeme poskládat jakýkoliv tvar. V 2D grafice je jednoduché si představit, že za pomoci dvou trojúhelníků složíme čtverec. 3D grafika již není tak jednoduchá, obzvláště u složitějších objektů jako je koule, která se dá také poskládat z trojúhelníků. Čím více jich použijeme, tím bude povrch koule hladší.

Na objektech v prostoru můžeme provádět různé transformace, např. rotace, změna měřítka, změna polohy. Stejným způsobem zacházíme i s objekty v reálném světě. Každá transformace ovlivňuje body (vertexy) daného objektu, u kterých se mění jejich souřadnice. Předměty, s kterými každý den pracujeme nebo je jen pozorujeme, jsou z určitého materiálu. I ve virtuálním světě lze objektům nastavit jejich vlastnosti materiálu a vzhled textury, která dodává povrchu reálný vzhled.

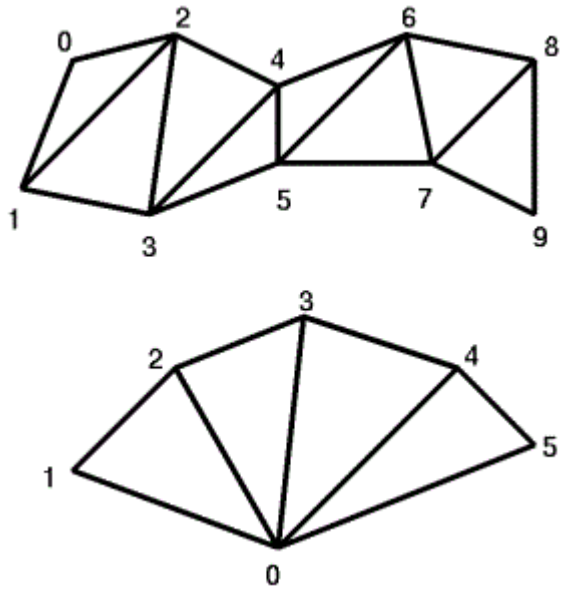
Tento editor se tedy zaměřuje na práci s objekty v prostoru. Hlavními funkcemi je transformace jednotlivých objektů, vertex bodů a plošek. Každý objekt má svůj materiál složený z různých složek (ambientní, difúzní, spekulární), nebo lze nastavovat a měnit texturu objektu. Pro lepší zobrazení objektů v prostoru je možno přidávat bodová světla, mazat nebo upravovat jejich pozici. Zobrazení celé scény nám zprostředkovává kamera, u níž lze nastavit i vícenásobné zobrazení.

2 Reprezentace těles

Pod pojmem těleso si můžeme představit objekt reálného světa, tedy hmotný předmět, který v prostoru zaujímá určitý objem. Těleso lze také považovat za množinu bodů v trojrozměrném prostoru. Je chápáno jako spojitý útvar tvořící jeden celek i s možnými otvory. [5]

2.1 Síť trojúhelníků

Trojúhelník má dobré vlastnosti, které se dají v grafice výhodně využít. Tou nejpodstatnější vlastností je, že trojúhelník vždy leží v jedné rovině a lze velmi lehce spočítat jeho normálu a těžiště. Síť tzv. triangle mesh je množinou trojúhelníků , které mají společné hrany. Datová struktura objektu popsaná tímto způsobem bývá rozdělena do dvou logických částí. První z nich je geometrická, která zaznamenává souřadnice vrcholů. Druhou je topologická, uchovávající údaje o vrcholech, které společně tvoří trojúhelník, ale také může obsahovat informace o tom, které trojúhelníky spolu sousedí. Toto rozdělení je praktické například pro geometrické transformace. Zde se pracuje jen s geometrickou částí a pouze se přepočítají souřadnice vrcholů. Snížení náročnosti zpracování dat v grafickém procesoru, neboli zmenšení počtu operací nad jednotlivými vrcholy, dosáhneme seřazením trojúhelníků do řady tak, aby odpovídaly pruhu trojúhelníků znázorněném na obr. 2.1 nahoře. Ten nám ukazuje, že se každý vrchol zpracuje pouze jednou. Každý vrchol tvoří s předchozími dvěma jeden trojúhelník. Podobným způsobem lze také vytvořit vějíře trojúhelníků, viz. obr. 2.1 dole. [5]

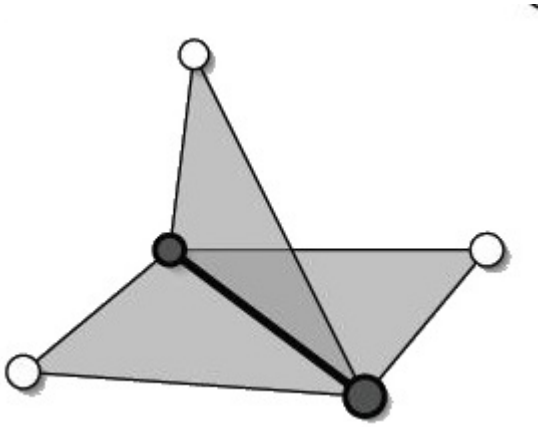


Obrázek 2.1: Pruh trojúhelníků(nahoře) a vějíř trojúhelníků(dole)

Síť trojúhelníků má i své nevýhody, jako je například mapování textur, u čehož je potřeba určit explicitní vztah každého vrcholu k obrázku textury.

2.2 Hraniční reprezentace

Tento způsob reprezentace je založen na popisu hranice tělesa. Ta je popsána množinou hraničních bodů, jako i v reálném světě znázorňujeme tělesa podle jejich obrysu (podmnožina hranic). Jejich hranice je popisována základními prostorovými prvky, jimiž jsou body, úsečky a části rovinných ploch. Tělesa, jejichž každá hrana spojuje dvě plochy a zároveň neprotínají jiné, nazýváme manifold. Takovéto objekty mají i svůj protějšek a tím je tzv. nonmanifold znázorněný na obr. 2.2.



Obrázek 2.2: Nonmanifold [13]

Zvýrazněná hrana je průsečnicí tří ploch. Tento spoj si umíme představit jako nekonečně tenkou přímku, ale ve skutečném světě nedokážeme vyrobit nekonečně tenké vlákno ani dokonalý bodový svár, kterým bychom tyto plochy propojily. Ve skutečnosti by na tom místě musely být dvě hrany. Popis povrchu těles pouze pomocí vrcholů a hran je jedním z nejjednodušších způsobů hranové reprezentace. Tímto zobrazením dostaneme tzv. drátový model tělesa. V seznamu vrcholů jsou uloženy souřadnice jednotlivých bodů a v seznamu hran má každá dva ukazatele na vrcholy. Prosté vykreslení hran nedokáže poskytnou veškeré informace o vzhledu objektu, ale pro přehlednější a lepší prozkoumání vnitřní struktury tělesa je vhodný. Lze rozšířit strukturu o vrstvu ploch. V takovém případě už jde o jednoduchou plošnou reprezentaci, která bere v potaz hranici tvořenou plochami. Ty mohou být skládány z trojúhelníků nebo polygonů. První varianta je jednodušší, jelikož stačí vytvořit pole, jehož položka vždy obsahuje odkaz na tři vrcholy. Pořadí vrcholů by mělo být uchováváno podle orientace plochy a řadit se proti směru pohybu hodinových ručiček. Díky tomu lze odvodit normálový vektor, který směruje ven z tělesa. [5]

3 Transformace

Transformace jsou jedny z nejpoužívanějších operací v počítačové grafice. Můžeme je klasifikovat na lineární a nelineární. Mezi lineární transformace patří rotace, posunutí, změna měřítka, zkosení a operace vzniklé jejich kombinacemi. S nelineárními transformacemi se v grafice setkáváme u funkcí, které provádí složitější změny tvaru objektů, např.: deformace prostorových modelů. Jsou aplikovány buď na bod nebo na objekt (všechny body objektu). Zvláštní transformací je projekce, která převádí objekty z vícerozměrného prostoru do prostoru o méně rozměrech. S projekcí se setkáme například při převodu z trojrozměrné scény do roviny obrazu. Transformace jsou aplikovány na jednotlivé souřadnice objektu, čímž mění svou polohu. [7]

Homogenní souřadnice

Homogenní souřadnice bodu v 3D s kartézskými souřadnicemi [x, y, z] je uspořádaná čtveřice [X, Y, Z, w] pro kterou platí x = X/w, y = Y/w, z = Z/w. Souřadnice w je nazývána váhou bodu. Reprezentace bodů pomocí homogenních souřadnic je výhodou pro zjednodušení výpočtu transformací. Nejčastěji používané lineární transformace jsou vyjadřovány pomocí jedné matice s homogenními souřadnicemi. Způsob vyjádření transformací pomocí jedné matice je výhodný pro jejich implementaci, protože se dají využívat knihovny pro práci s maticemi. Více transformací se dá složit do jedné matice jejich násobením mezi sebou. Jelikož jde o násobení matic, záleží na pořadí. Inverzní transformace je vyjádřena inverzní maticí. U lineárních transformací je hodnota w = 1.

Bod, který chceme transformovat, má matici P(x, y, z, w), kde x, y, z jsou souřadnice, w je 1 pokud je to bod, 0 pokud je to vektor. Transformaci prezentuje tzv. transformační matice. [7]

Transformační matice

Transformační matice v trojrozměrném prostoru má tvar 4x4. Vytvoříme ji vzájemným roznásobením matic jednotlivých transformací. Pozor, záleží na pořadí násobení. Je rozdíl v tom, zda nejdříve s objektem posune a poté otočíme kolem středu souřadného systému, nebo zda nejdříve otočíme a následně posuneme. Pokud se mají transformace provést v pořadí A1, A2, tak výslednou transformační matici získáme vztahem T = A2 * A1. [7][5]

3.1 Translace

Translace neboli posunutí ve 3D prostoru je určeno vektorem posunutí

$$\vec{p} = (X_t, Y_t, Z_t)$$

Tvar matice pro posunutí vypadá následovně:

$$T = T(X_t, Y_t, Z_t) = \begin{bmatrix} 1 & 0 & 0 & X_t \\ 0 & 1 & 0 & Y_t \\ 0 & 0 & 1 & Z_t \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

a její inverzní matice:

$$T^{-1} = T(-X_t, -Y_t, -Z_t) = \begin{bmatrix} 1 & 0 & 0 & -X_t \\ 0 & 1 & 0 & -Y_t \\ 0 & 0 & 1 & -Z_t \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Příklad posunutí o 3 po ose x:

$$T = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

[7][5]

3.2 Změna měřítka

Transformační matice pro změnu měřítka vypadá následovně:

$$S(X_s, Y_s, Z_s) = \begin{bmatrix} X_s & 0 & 0 & 0 \\ 0 & Y_s & 0 & 0 \\ 0 & 0 & Z_s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

a její inverzní matice:

$$S^{-1}(X_s, Y_s, Z_s) = S(1/X_s, 1/Y_s, 1/Z_s) = \begin{bmatrix} 1/X_s & 0 & 0 & 0 \\ 0 & 1/Y_s & 0 & 0 \\ 0 & 0 & 1/Z_s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Koeficienty $s_x \neq 0$, $s_y \neq 0$ a $s_z \neq 0$, určují změnu ve směru příslušné souřadnicové osy.
Příklad změny měřítka na ose y:

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

[7][5]

3.3 Rotace

Rotace, neboli otáčení, reprezentuje matice pro R_x otáčení kolem jednotlivých os. Matice znázorňuje rotaci kolem osy x o úhel α.
Transformační matice pro rotaci kolem osy x:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformační matice pro rotaci kolem osy y:

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformační matice pro rotaci kolem osy z:

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Složením několika dílčích transformací kolem os x, y, z lze v prostoru realizovat otočení kolem obecné osy. Taková transformace se hledá složitě. Jednoduchým způsobem je využití Rodriguesovy formule. Vytvoříme transformační matici, která převádí rotační úlohu na promítání a skládání několika vektorů.

Předpokládejme, že osa otáčení je určena počátkem souřadnicového systému O a jednotkovým vektorem $\vec{a}$. Polohový vektor transformovaného bodu X je kolem této osy otáčen o úhel α a výsledný polohový vektor je označen $\vec{x}'$. Tuto rotaci popíšeme vztahem:

$$\vec{x}' = \cos\alpha \cdot \vec{x} + (1 - \cos\alpha)(\vec{a} \cdot \vec{x}) + \sin\alpha(\vec{a} \times \vec{x})$$

Rotaci bodu X lze přepsat do maticového tvaru s využitím definic vektorového a skalárního součinu, matice I je jednotková matice (identita).

$$X' = R(\vec{a}, \alpha) \cdot X$$

$$R(\vec{a}, \alpha) = \cos\alpha \cdot I + (1 - \cos\alpha) \cdot \begin{bmatrix} a_x^2 & a_x a_y & a_x a_z & 0 \\ a_x a_y & a_y^2 & a_y a_z & 0 \\ a_x a_z & a_y a_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \sin\alpha \cdot \begin{bmatrix} 0 & -a_z & a_y & 0 \\ a_z & 0 & -a_x & 0 \\ -a_y & a_x & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

[7][5]

4 Promítání

Promítání je transformace, která převádí trojrozměrné objekty na dvourozměrné. Tím dochází k možnému zkreslení názoru na skutečný tvar z pohledu pozorovatele. Promítání používá základní pojmy jako je promítací paprsek a průmětna.

- Promítací paprsek je polopřímka vycházející z 3D bodu scény a dopadá na průmětnu.
- Průmětna je teoreticky neomezená plocha (ve skutečnosti výřez) v prostoru, na kterou dopadají promítací paprsky. V místě dopadu vzniká průmět, neboli obraz bodu (objektu).

Rovinné promítání dělíme na do dvou základních skupin. První je paralelní tzv. rovnoběžné promítání, druhá perspektivní neboli středová. Tyto třídy rozlišujeme podle charakterizování směru paprsků. U paralelního mají všechny paprsky stejný směr, na rozdíl od perspektivního,kde vycházejí z jediného bodu. Úkolem promítání je nalézt jednu nebo více transformačních matic. Souřadnice promítnutého bodu na průmětnu vypočítáme pomocí vztahu:

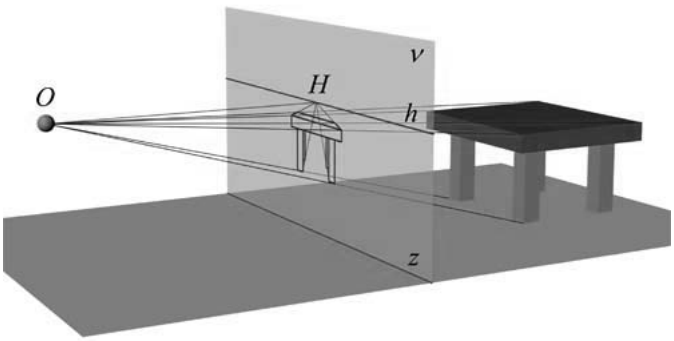
$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ w_p \end{bmatrix} = T_p \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

kde T_p je transformační matice 4×4 promítání bodu $[x, y, z, 1]$. Výsledné souřadnice promítnutého bodu jsou $P_p = [x_p, y_p, z_p, w_p]$. [5]

Projekční matice Obraz trojrozměrné scény je promítán na obrazovku počítače do dvourozměrného prostoru. Pro tuto transformaci se používá právě projekční matice, která transformuje vrcholy z trojrozměrného do dvourozměrného prostoru. Její tvar je 4×4 a při společném vynásobení s vrcholem ve scéně získáme pozici vrcholu na obrazovce.[11]

4.1 Středové promítání

Vytváří obrazy, které jsou podobné těm, které vidí lidské oko. Vzdálenost objektů od středu promítání ovlivňuje velikost průmětů, nebo-li vzdálenější objekty mají menší průměty. Promítací paprsky procházejí jedním bodem a tím je střed promítání, proto se nezachovává rovnoběžnost. Právě to je charakteristické pro středové promítání. I když jsou úsečky(hrany) objektu v trojrozměrném prostoru rovnoběžné, tak jsou zobrazovány do dvourozměrného prostoru jako mimoběžné. Výjimkou mohou být úsečky(hrany), které leží v rovině rovnoběžné s průmětnou. Na obrázku 4.1 je znázorněno středové promítání obrazce nacházejícího se za průmětnou.



Obrázek 4.1: Středové promítání [14]

Objekt je tedy zobrazen ve zmenšené velikosti, než je ta reálná. Pokud by se nacházel na průmětně, tak by byla velikost totožná s reálnou. Avšak v případě postavení tělesa před průmětnou bude jeho velikost větší. Průmětna může mít jakoukoliv polohu a tak jsou rozlišeny tři druhy perspektivy, které odpovídají orientaci průmětny vůči souřadným osám.

1. Jednobodová perspektiva - vzniká v případě, kdy průmětna protíná pouze jednu souřadnicovou osu a tak všechny úsečky, které jsou na ni kolmé, míří do jediného bodu, který nazýváme hlavní úběžník. Ukázka jednobodové perspektivy je na obrázku 4.2 (vlevo)

2. Dvoubodová perspektiva – vytvoří se v případě, že průmětna protíná dvě souřadnicové osy a hrany osově orientovaných objektů jsou směřovány do dvou hlavních úběžníků. Na obrázku 4.2 (uprostřed) je ukázka dvoubodové perspektivy.

3. Trojbodová perspektiva - protažením hran, které jsou osově orientované, získáme tři hlavní úběžníky. Poslední část obrázku 4.2 (vpravo) nám znázorňuje příklad trojbodové perspektivy.

Obrázek 4.2: Perspektiva jednobodová (vlevo), dvoubodová (uprostřed) a trojbodová (vpravo) [14]

Uvažujeme-li střed promítání S a průmětnu ϱ, přičemž platí, že střed promítání neleží na průmětně, tak obrazem středového promítání libovolného bodu A, který není roven S, je $A' \in \varrho$. Středem promítání je tedy bod $S = (m, n, p, 1), p \neq 0$ a průmětnou je půdorysna $\pi = (x, y)$. Potom výslednou transformační maticí je

$$G = \begin{bmatrix} p & 0 & 0 & 0 \\ 0 & p & 0 & 0 \\ -m & -n & 0 & -1 \\ 0 & 0 & 0 & p \end{bmatrix}$$

a středovým průmětem bodu $A = (x_A, y_A, z_A, 1)$ je

$$A' = A \cdot G = \left(\frac{px_A - mz_A}{p - z_A}, \frac{py_A - nz_A}{p - z_A}, 0, 1 \right)$$

[5][8].

4.2 Rovnoběžné promítání

Rovnoběžné promítání se především využívá ve strojírenství. Určuje ho směr promítání a průmětna. Zanechává si rovnoběžnost a tak průmět, který leží v rovině s průmětnou je shodný s promítaným útvarem. Rovnoběžným průmětem objektu je objekt, který získáme rovnoběžným průmětem všech bodů promítaného objektu.

- Průmět bodu je bod.
- Přímka rovnoběžná se směrem promítání bude zobrazena jako bod, ale přímka která je různoběžná, bude vyobrazena jako přímka.
- Rovinu rovnoběžnou se směrem promítání bude vyobrazena přímkou (průsečnice promítané roviny s průmětnou). V případě různoběžné roviny je to celá průmětna.

Podle úhlu, které svírají směrové paprsky vůči průmětně rozlišujeme, zda jde o pravoúhlé nebo kosoúhlé rovnoběžné promítání.[8]

4.2.1 Pravoúhlé promítání

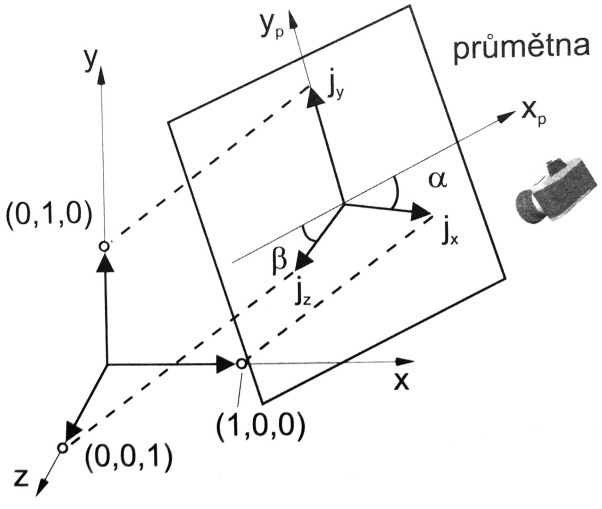
Již podle názvu je jasné, že pravoúhlé bude na průmětnu kolmé. V případě, že průmětna zaujímá obecnou polohu vůči souřadnému systému a promítání je pravoúhlé, mluvíme o tzv. axonometrii. Například u technických výkresů se nejčastěji používá tzv. Mongeovo promítání.[8][5]

Mongeovo promítání Skupina promítání ve směru hlavních os do průměten v hlavních rovinách, která zahrnuje nárys a podle potřeb bokorys, půdorys (pohled shora), spodní pohled a pohled zezadu, se nazývá Mongeovo promítání. [5] Jde o nejjednodužší rovnoběžné promítáním, které zobrazuje objekty na dvě vzájemně kolmé průmětny. Příklad transformačních matic podle hlavních rovin:

$$T_{xy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, T_{xz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & y_0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, T_{yz} = \begin{bmatrix} 0 & 0 & 0 & x_0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

[5]

Axonometrie Průmětna není rovnoběžná s hlavními osami (protíná dvě nebo tři). Tři úsečky v rovině, které mají společný jeden krajní bod a které neleží v přímce, lze pokládat za rovnoběžný průmět tří vzájemně kolmých a stejně dlouhých úseček, které mají jeden krajní bod společný. V souladu s tím je axonometrie často definována pěti hodnotami $j_x, j_y, j_z, \alpha, \beta$, kde j_x, j_y, j_z jsou průměty jednotek na osách x, y, z. Úhly α, β svírají promítnuté osy j_x, j_z s kolmicí na průmět osy j_y.[5]



Obrázek 4.3: Axonometrie[5]

Speciální druhy axonometrií:

- isometrie: $j_x = j_y = j_z, \alpha = \beta$
- dimetrie: $j_x = j_y, \alpha = \beta$
- trimetrie: $j_x \neq j_y \neq j_z$
- technické kosoúhlé promítání: $j_y = j_z = 1, \beta = 0$

4.2.2 Kosoúhlé promítání

Průmětnou zpravidla bývá bokorysna $\mu = (y, z)$ a směr promítání, který s ní svírá jiný úhel než $90°$ nebo $0°$. Nejčastěji používaným kosoúhlým promítáním je buď tzv. kavalír nebo tzv. kabinet. Pro svíraný úhel $45°$ mezi průmětnou a směrem paprsků mluvíme o kavalírním promítání. Rovnoběžné i kolmé úsečky s průmětnou jsou zobrazovány se stejnou délkou. Transformační matice této projekce:

$$T_{kav} = \begin{bmatrix} 1 & 0 & -\cos\beta & 0 \\ 0 & 1 & -\sin\beta & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

U kabinetního promítání, které svírá úhel $63.4°$, se úsečky kolmé na průmětnu zkracují na polovinu. Transformační matice této projekce:

$$T_{kab} = \begin{bmatrix} 1 & 0 & -\frac{\cos\beta}{2} & 0 \\ 0 & 1 & -\frac{\sin\beta}{2} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

[8][5]

4.3 Kamera

V grafice se kamera používá jako pozorovatel, a tak je zapotřebí ji nadefinovat některé vlastnosti. U kamery je důležité vědět její umístění v prostoru, orientaci průmětny, nadefinování směru a cíle pozorování. Kamera může snímat celý poloprostor před ní buď středovým nebo rovnoběžným promítáním a pořizuje snímky na průmětnu, která je kolmá ke hlavní optické ose kamery. Uvažujeme-li, že kamera má v prostoru světové souřadnice $[k_x, k_y, k_z]$ a cíl se souřadnicemi $[c_x, c_y, c_z]$, na který je namířena, tak můžeme určit směr pozorování $\vec{L} = (L_x, L_y, L_z, 0)$, který je rovnoběžný s hlavní optickou osou kamery.

$$\begin{bmatrix} L_x \\ L_y \\ L_z \\ 0 \end{bmatrix} = \begin{bmatrix} c_x \\ c_y \\ c_z \\ 1 \end{bmatrix} - \begin{bmatrix} k_x \\ k_y \\ k_z \\ 1 \end{bmatrix}$$

[5]

5 Textury

V dnešní době jsou textury neodmyslitelnou částí počítačové grafiky. Popisuje vlastnosti povrchu, které jsou důležité pro vnímání struktury, barvy a zvyšuje vizuální kvalitu objektu. Je mnohem snazší použít geometricky jednoduchý objekt se složitou texturou, než vytvářet složité geometrické detaily. Toho si můžeme všimnout hlavně u objektů, které jsou zobrazeny v krátké nebo velké vzdálenosti, takže rozdíl by byl těžce rozeznatelný. Povrch popisuje vlastnosti, jako je například jeho barva, kterou určuje koeficient difúzního odrazu. Nejčastěji se používá mapování difúzní složky materiálu. U povrchu lze také určit jeho odraz světla a můžeme dosáhnout až zrcadlového povrchu. Optickou změnu tvaru dokážeme ovlivnit pozměněním normálového vektoru, což nemá žádný vliv na tvar objektu. Povrch lze také zprůhlednit. Aplikace textury obnáší dva kroky: první je definice textury a druhý její mapování.

5.1 Mapování textur

Textura je běžným 2D obrázkem, ale musíme ho brát jako zdroj barevných informací pro rendrovací jednotku. Textura je nanášena jednotlivě na každý trojúhelník celé síťoviny objektu. Pro každý vrchol je nadefinována UV souřadnice textury. Proto se u textur jednotlivé grafické body nazývají texely. Každá textura má rozsah souřadnic od 0.0 do 1.0 v každém směru.

Mapování rovinné textury Mapování rovinné textury je jednou z nejjednodušší metod. Lze ji definovat funkcí $T(u, v)$ přiřazující bodům $[u, v]$ v rovině mapované veličiny, např.barvu.

$$T : D_T \rightarrow H_T, D_T \subset R^2$$

Proces nanášení textur využívá funkce inverzního mapování ($M(x, y, z)$), která každému bodu tělesa přiřadí bod z definičního oboru textury T.

$$M : D_M \rightarrow D_T, kde D_M \subset R^3 jsou body na povchu tělesa$$

Nanášení textury si lze představit tak, jako kdyby se objekt polepoval papírem. Volba funkce M by měla odpovídat tvaru tělesa, na který je textura nanášena, jinak by mohlo dojít k jejímu zkreslení.

Mapování válcové a kulové plochy Inverzní mapovací funkce pro válcovou a kulovou plochu je už složitější. Mapování se musí nadefinovat tak, aby textura pokryla celý plášť. Do výpočtu jsou tedy zapojeny i další veličiny jako je výška *h* a poloměr *r*. U takovýchto objektů se používají cylindrické souřadnice. V závislosti na poloměru a úhlu natočení vyjadřují souřadnice bodu, neboli $[x, y] = [r \cos \alpha, r \sin \alpha]$. Pomocí toho se dá odvodit mapovací funkce $M(x, y, z)$, která $\arccos$ vrací hodnoty v intervalu $< 0, \pi >$:

$$u = \begin{cases} \frac{1}{2\pi} \arccos \frac{x}{r}, & proy \leq 0 \\ 1- \frac{1}{2\pi} \arccos \frac{x}{r}, & proy > 0 \end{cases}$$

$$v = \frac{z}{h}$$

Textura může být větší než je mapovaný objekt, a tak nedojde k zobrazení některé její části. V opačném případě, kdy je textura menší, se buď ukončí, a nebo se opakuje v jednom nebo obou směrech.[5]

Mapování prostorové textury Takto lze modelovat tělesa, která lze přirovnat k tělesům vyřezaných z jednoho kusu materiálů, např. dřevo, mramor, apod. Vnitřní strukturu popisuje textura. Proces spočívá v aplikaci souřadnic $[x, y, z]$ texturovaného bodu k nalezení odpovídajícího bodu $[u, v, w]$ v textuře.[5]

Mapování prostředí Tato technika se také nazývá chromové mapování, jelikož modeluje zrcadlení textury. Odraz na povrchu je typický pro zrcadlové materiály. Závisí na poloze tělesa, na okolním prostředí, ale také na tom, kde se nachází pozorovatel. Je to efektivní technika, která se používá v rozsáhlých scénách, kde je potřeba využít odrazy.[5]

Nerovné textury Technika, která vytváří nerovnosti na povrchu. Dochází k tomu za pomoci modifikace normály každého pixelu povrchu tělesa. Ta ovlivní výpočet osvětlení a vzniká tím následně falešný stín. Samozřejmě povrch zůstává svým tvarem stejný. Ke změně normál se používá výšková mapa. Je to textura s 8b barevnou hloubkou tzv. bump map. Jedná se pouze o iluzi, takže deformace povrchu ve větší rozsahu takto vizualizovat nelze. [5]

6 Osvětlení

Osvětlení je jedním z nejdůležitějších faktorů při tvorbě realistické scény. Jde o vzájemnou interakci mezi světelným zdrojem a 3D objektem v prostoru. Barvy, které jsou vidět v reálném světě, jsou výsledkem tohoto působení složeného minimálně ze tří aspektů (světelné zdroje, materiál a pozorovatel). Ze světelného zdroje dopadá světlo určitého spektra na povrch. Zde se část vstřebává a zbytek se rozptyluje nebo odráží. Úhel odrazu je závislý na úhlu dopadu a kolmici k povrchu. Velikost rozptýlení ovlivňuje materiál povrchu a jeho hladkost. Barevné spektrum odraženého světla určuje dopadající spektrum a absorpční vlastnosti materiálu. Síla závisí na poloze a vzdálenosti od zdroje a pozorovatele, ale také na materiálu. Osvětlování modelů rozdělujeme do dvou skupin:[5]

Lokální osvětlení Jsou to pouze přímé zdroje světla. Barvu povrchu ovlivňují vlastnosti povrchu a přímá světla.

Globální osvětlení V reálném světě existují nepřímé zdroje světla, jako jsou odrazy od jiných objektů nebo prostředí.

Světelné zdroje v počítačové grafice Světelný zdroj je obecně jakýkoliv objekt, a proto můžeme záření emitovat, ale i odrážet. Je charakterizován emisním spektrem, tedy funkcí, která udává světelný výkon pro každou vlnovou délku viditelného spektra.[5]

Bodový světelný zdroj Vyzařuje světlo rovnoměrně a se stejnou intenzitou do všech směrů. Je určen svou intenzitou a polohou.

Rovnoběžný světelný zdroj Lze chápat jako nekonečně veliký rovinný zdroj v nekonečné vzdálenosti. Paprsky dopadají rovnoběžně a napodobují například sluneční svit.

Plošný zdroj Má nekonečnou plochu a vyzařuje paprsky do předního poloprostoru všemi směry. Oproti bodovému světlu způsobuje stíny s neostrou hranou, tzv.polostíny.

Reflektor Světlo, které je závislé na směru. Je tedy určen svou polohou a směrem, kterým září. Nejvíce vyzařuje ve směru osy vyzařování. Geometricky ho lze popsat jako kužel.

[5]

6.1 BRDF funkce

Funkce, která určuje, jaká bude odrážená intenzita v závislosti na pozici světla a pozorovatele. Každý materiál má odlišnou odrazivost. Barva objektů je ovlivněna spektrální charakteristikou dopadajícího světla, ale hlavně vlastnostmi povrchu. Především jaké vlnové délky a v jakém směru odráží. Pro tento model jsou definovány některé předpoklady. Nebude brán v potaz jev fosforescence, tj. "nabití"materiálu světlem, takže světlo

se okamžitě odráží a nevstřebává. Dalším ignorovaným jevem je fluorescence. Tudíž vlnová délka dopadajícího světla se nijak nemění a odráží se na stejné délce. BRDF dodržuje tzv. Helmholtzův pricip preciprocity, který říka, že i když změníme směr dopadu a odrazu světelného paprsku, tak BRDF v daném bodě zůstane stále stejné.

$$f_r(x, \vec{w_r}, \vec{w_i}) = f_r(x, \vec{w_i}, \vec{w_r})$$

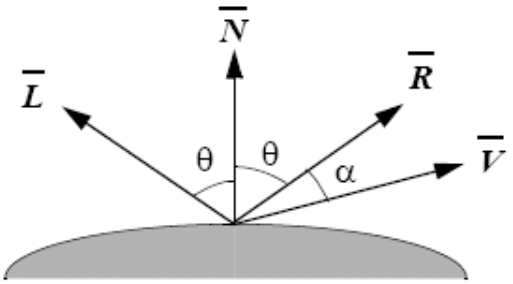
Tento model dodržuje stejný směr odrazu a dopadu i v opačném směru. To tedy znamená, že pokud je vstupním směrem $\vec{w_i}$ a výstupním je $\vec{w_r}$, tak pokud se obrátí směr a vstupním směrem je $\vec{w_r}$, výstupním směrem se stane $\vec{w_i}$. BRDF není nikdy záporná tedy: $f_r(x, \vec{w_r}, \vec{w_i}) \geq 0$. Je anizotropní. To znamená, že odraz nezáleží na vstupní a výstupním směru a bodu x, ale také na na natočení povrchu kolem normálového vektoru k tomuto povrchu. BRDF by tedy měla mít tvar $f_r(x, \vec{w_r}, \phi, \vec{w_i})$, kde ϕ určuje úhel natočení bodu. Řídí se také zákonem zachování energie, takže energie nemůže vzniknout ani zaniknout při dopadu. Pouze se může měnit z jednoho druhu na jiný. Pro vyjádření vlastností materiálů se v BRDF využívá tzv. odrazivost $\rho(x)$. Jde o poměr odraženého toku $\Phi_r(x)$ v bodě x k dopadajícímu světelnému toku $\Phi_i(x)$ v témže bodě:

$$\rho(x) = \frac{d\Phi_r(x)}{d\Phi_i(x)}$$

Hodnota odrazivosti se pohybuje v intervalu $< 0, 1 >$. V případě, že je hodnota $\rho(x) = 1$, dochází k úplnému odrazu světla. Nic se nepohlcuje ani neprochází materiálem.

6.2 Phongův osvětlovací model

Pro výpočet odráženého světla z povrchu nějakého objektu využíváme právě tento model. V roce 1975 ho navrhl Bui-Tuong Phong. V počítačové grafice slouží k vykreslování. Na obrázku č.6.1, který znázorňuje geometrii odrazu podle Phongova modelu.



Obrázek 6.1: Geometrie odrazu [12]

Znázornění odrazu na obrázku č.6.1 směrem k pozorovateli $\vec{V}$ nám určuje směr dopadajícího světla $\vec{L}$, normálový vektor v místě dopadu $\vec{N}$ a zrcadlově odražený paprsek$\vec{R}$. Phongův osvětlovací model rozlišuje tři druhy odrazů světla od materiálu a z nich se složí jejich součtem výsledný odraz. Rozdělujeme je na spekulární (zrcadlový), difúzní a ambientní složku. [6]

Ambientní odraz Vzniká odrazem okolního neboli tzv. globálního světla. Okolní světlo zajišťuje, aby povrchy odvrácené od zdrojů nebyly zcela černé. Čím větší je hodnota, tím je scéna světlejší. Tuto složku značíme I_a a je určena vztahem:

$$I_a = I_a r_a$$

I_a označuje intenzitu okolního osvětlení (bývá pro celou scénu stejná) a $r_a \in (0, 1)$ je odrazivý koeficient materiálu objektu určující schopnost povrchu odrážet okolní světlo. Pro objekty odvrácené od všech zdrojů je tato složka velice důležitá, protože tak nejsou zobrazovány zcela černé. [5]

Difúzní odraz Udává intenzitu světla, které se od tělesa odráží rovnoměrně, tedy do všech směrů. Tato složka již vytváří trojrozměrný vzhled objektu. Je značena I_d a popsána vztahem:

$$I_d = I_L r_d(\vec{l} \cdot \vec{n})$$

I_L reprezentuje barevné složení dopadajícího paprsku a r_d je koeficient difúzního odrazu. Vektor $\vec{n}$ je normála povrchu v místě dopadu paprsku a $\vec{l}$ příchozí směr světla k povrchu. Čím je větší I_d, tím je směr dopadu bližší k normále. Vztah má smysl pouze pokud $\vec{l} \cdot \vec{n} > 0$. V opačném případě je povrch odvrácen od zdroje. [5]

Spekulární odraz Spekulární odraz není ideálním zrcadlovým odrazem, ale spíše lesk povrchu. Tato složka se značí I_s a je vyjádřena vztahem:

$$I_s = I_L r_s(\vec{v} \cdot \vec{r})^h$$

I_L reprezentuje barevné složení dopadajícího paprsku a r_s je koeficient zrcadlového odrazu $0 \leq r_s \leq 1$, který určuje míru zastoupení odražené zrcadlové složky světla v celkově odraženém světle.[5] Vektor $\vec{v}$ je jednotkový vektor pohledu a $\vec{r}$ vyjadřuje ideální směr zrcadlového odrazu. Koeficient h je skalární a vyjadřuje ostrost zrcadlového odrazu. Udává se v rozmezí $< 1, \infty)$. S rostoucím h jsou odlesky na zobrazovaném tělese menší a ostřejší, dokonalé zrcadlo má $h = \infty$. [5]

7 Scéna

Množinu prostorových objektů společně s informacemi potřebnými k zobrazení této množiny nazýváme scénou. Sice nezpracovává geometrická data objektů, které reprezentují tvar objektu, ale stará se o transformaci objektů do jejich polohy. Nacházejí se tam tedy nezobrazované objekty (kamery a osvětlení scény) a zobrazované objekty (jejich tvar, barevné vlastnosti a textury). Vlastnosti kamer jsou popsány v kapitole 4.3 a osvětlení v kapitole 6. Informace o objektech lze uchovávat v souborech s formátem .obj, které slouží k uchovávání informací o 3D objektech. Do scény tedy lze naimportovat najednou více objektů z jednoho takovéhoto souboru, ale lze i objekty ve scéně vyexportovat tak, abychom zachovali informace o objektech a jejich umístění ve scéně.[5]

7.1 Souborový formát .obj

Objektové soubory mohou být ve formátu ASCII (.obj) nebo binárním formátu (.mod). Polygonální geometrie používá vrcholy, linie a plochy pro definování geometrie objektů. Tento formát se tedy používá k překladu geometrických informací pro jiné aplikace.

Základními prvky struktury souboru

Název objektu (o). Vlastnosti vrcholů:

- pozice vrcholu (v) - v x y z, kde x, y a z jsou souřadnice v prostoru
- mapování textury (vt) - u v, kde u je horizontální směr textury a v je vertikální
- normála vrcholu (vn) - vn i j k, kde i, j a k definují normálový vektor

příklad definování vlastností vrcholu:

v -1.0 1.0 0.0
v -1.0 -1.0 0.0
v 1.0 -1.0 0.0
v 1.0 1.0 0.0
vt 0.21 3.59
vt 0.0 0.0
vt 1.0 0.0
vt 0.5 0.5
vn 0.0 0.0 1.0
vn 0.0 0.0 1.0
vn 0.0 0.0 1.0
vn 0.0 0.0 1.0

Základními prvky:

- body (p) - p v1 v2 v3 $\cdots$

- linie (l) - l v1/vt1 v2/vt2 v3/vt3 · · ·
- plocha (f) - f v1/vt1/vn1 v2/vt2/vn2 v3/vt3/vn3 · · ·

příklad zápisu krychle bez textur a normál:
o kostka
v 0.0 2.0 2.0
v 0.0 0.0 2.0
v 2.0 0.0 2.0
v 2.0 2.0 2.0
v 0.0 2.0 0.0
v 0.0 0.0 0.0
v 2.0 0.0 0.0
v 2.0 2.0 0.0
f 1 2 3 4
f 8 7 6 5
f 4 3 7 8
f 5 1 4 8
f 5 6 2 1
f 2 6 7 3

V případě plochy bez textury, ale s normálou bude mít zápis tvar *f 1//1 2//2 3//3*.[9]

8 Technologie použité při implementaci

8.1 HTML5

Hypertext Markup Language (HTML) je formát dokumentu užívaný pro webové stránky. Standard HTML definuje tagy a kódy použité pro určení uspořádání textu, fontů, stylů, obrázků a dalších prvků, které tvoří vzhled webové stránky. [1]

Historie

Jazyk HTML byl vyvinut v roce 1989 jako součást projektu WWW, který vznikal v CERNu. Za jeho zrodem stáli Tim Berners-Lee a Robert Caillau. Tento projekt měl v úmyslu umožnit vědcům komunikovat a sdílet výsledky po celém světě. O dva roky později byla vyvinuta první verze, která zvládala zvýrazňování textu, přidávání odkazů a obrázků, strukturovat obsah do různých logických úrovní. Požadavky uživatelů vzrůstaly a tak prohlížeče začaly obohacovat jazyk HTML o další prvky.

Vývoj přinesl HTML 2.0, který definoval práci s formuláři. Standard byl vydán v roce 1994.

Další verze byla pro svou dobu příliš velkým pokrokem, protože nikdo nedokázal vytvořit prohlížeč, který by dokázal podporovat HTML 3.0.

Konsorcium W3C rozhodlo o vzniku nového standardu HTML 3.2 tím, že rozšířili HTML 2.0. Tento krok přinesla práci s tabulkami, lepší formátování a volnější výběr druhů písma. Také se objevila podpora Java appletů. Rozšíření, která vznikla nad rámec HTML 3.2 nebyla podporována některými prohlížeči, ale jejich používání pomohlo k rychlejšímu zařazení do standardů. Mezi tato rozšíření patřily například skriptovací jazyky jako JavaScript.

Koncem roku 1997 byl vydán nový standard HTML 4.0, který přinesl začlenění skriptovacích jazyků a používání konstrukcí jako jsou rámy. Verze byla nahrazena novou 4.01. Ta opravila chyby z minulé a již plně podporovala kaskádové styly (CSS).

Dalším krokem ve vývoji se stala XHTML 1.0, která byla přenesena do standardu XML tak, aby vyhovovala normě XML. Jde o uzavírání tagů, jejich psaní malými písmeny a povinné hodnoty atributů. Další verze se staly slepou uličkou, protože je výrobci prohlížečů ignorovali.

O novém standardu se začalo uvažovat v roce 2008. Navazuje na HTML 4.01 a přidává nové užitečné funkce. Ten byl vydán 28. října 2014.[2]

Novinky v HTML 5

HTML5 se spíše stává prostředím pro vytváření webových aplikací, než pouhým jazykem pro statické stránky. Přináší jiný pohled na to, jak se má s webem pracovat. Přichází s novými funkcemi a jsou přidány i nové prvky, např.: article, section, header a další.

Nejdůležitější nové funkce:

- Application Cache - selektivní cachování webových stránek, obrázků, scriptů ...
- Audio/video - nativní podpora pro přehrávání vybraných formátů audia a videa.
- Data Storage - key-value uložiště lokálních dat vázaných na doménu.
- Drag & Drop - podpora pro drag & drop jedank v rámci prohlížeče, jednak drag & drop objektů pocházejících mimo prohlížeč.
- File API - přímý přístup k souborovému systému hosta.
- Graphics - podpora kreslení na Canvas a podpora vektorového jazyka SVG.
- History - pokročilá práce s historií prohlížeče.
- Semantics - nové sémantické elementy pro lepší identifikaci významů tagů.
- WebSockets - umožnění plně duplexní asynchronní komunikace mezi prohlížečem a serverem.
- WebWorkers - paralelní výpočty v JavaScriptu.

[2]

WebSockets

Je označováno jako jeden z největších kroků dopředu v HTML5. Většina webových aplikací v dnešní době potřebuje k provozu dlouhotrvající obousměrnou komunikaci. HTTP funguje v half-duplex režimu. Server odpovídá pouze na dotazy ze strany klienta. HTML5 využívá sokety, které simulují full-duplex režim, který umožňuje posílat zprávy na server a zároveň je i přijímat. Veškerá komunikace probíhá nad protokolem TCP přes standardní port 80 (HTTP). Jsou široce použitelné. Sokety mají své vlastní označení protokolu *ws*. V případě užití zabezpečené komunikace SSL je protokol označován *wss*. WebSocket používá ke komunikaci metody a události. Pro zasílání informací na server používá metodu *send*, která předává UTF-8 řetězec. V případě otevření spojení se serverem se vyvolá událost *onopen*. Události jsou vyvolávány také při zavření socketu (*onclose*) a nebo obdržení zprávy ze serveru (*onmessage*). U poslední zmíněné události (*onmessage*), je libovolné funkci *function(e) { ··· }*, předána příchozí zpráva v parametru *e.data*.

8.2 CSS3

Cascading Style Sheets (CSS)je jazyk pro popis prezentace webových stránek. Umožňuje přizpůsobit vzhled stránek na různých zařízeních, jako jsou velké nebo malé obrazovky, tiskárny, mobilní zařízení, atd.[3]

Historie

První specifikace CSS stylů se na svět dostala v roce 1996 a jejich tvůrcem bylo konsorcium W3C. V té době šlo stránky graficky upravovat dost obtížně, a tak příchod CSS stylů byl zlomovým okamžikem. CSS1 umožnilo nastavovat písma, barvy, pozadí, text, boxy, klasifikační vlastnosti a pozicování. Grafických efektů na stránkách se dosahovalo používáním velkých obrázků, protože podpora CSS stylů u prohlížečů nebyla příliš rozšířená.

Více jak jeden rok trvalo než se objevila nová úroveň kaskádových stylů a to CSS2. Kaskádové styly stále neměly velkou podporu u tvůrců webových stránek. CSS2 jsou kompatibilní se starší verzí a přidávají nové vlastnosti. Jednou z novinek je podpora více médií a zařízení se stránkovým výstupem. Dalším vylepšením jsou například: vlastnosti pro hlasové syntetizátory, lepší práce s fonty, pokročilejší formátování tabulek, řízení pozic, podstatné rozšíření selektorů a automatické číslování.

Dokončení nejnovějších CSS3 stylů je předpokládáno na rok 2015. Přestože stále není dokončená, mnohé prohlížeče velkou část této technologie již podporuje. Třetí verze zavádí specifikace tzv. modulů, které řeší jednu věc. Tento styl zjednodušuje věci, které se dříve zpracovávali pomocí JavaScriptu, jako jsou animace, ale také třeba zaoblení rohů, přechody nebo více obrázků na pozadí.[3]

Nejzajímavější novinky v CSS3

- Animace - první nativní webová animace. Všechny způsoby, které byly doposud využívány k tvorbě animací jsou buď technologie Silverlight, Flash, apod. a nebo se využívá JavaScript.
- Transition - animace přechodu má css vlastnost, kterou animuje. Nelze aplikovat úplně na všechny CSS vlastnosti.
- Funkce calc() - jednoduché výpočty pomocí této funkce.
- Pokročilé selektory - nové typy selektorů, jako například hvězdičkový, typový, atributový, kořenový, n-tý potomek, odkazový, atd.
- Counter - nejzajímavějším způsobem použití je například číslování nadpisů.
- Gradients - barevné přechody lze využít k vykreslení kde se donedávna používal externí obrázek. Novým přechodem je třeba Lineární nebo radiální.
- Vlastní fonty - dnes je to již standardní technika se skoro plnou podporou v prohlížečích.
- Box sizing - změna způsobu počítání šířky a výšky elementu.
- Border images - rámeček vykreslený obrázkem dává možnost vykreslit vlastní rámeček místo nativního.

- Media queries - podmíněná zobrazení pro média nastavuje styly pro jednotlivá média zvlášť.
- Multiple backgrounds - vrstvení více obrázků nebo barev na pozadí jednoho elementu. Syntaxe vlastnosti background je snadná, protože jednotlivé vrstvy jsou odděleny čárkou.
- Multi-column layout - tento modul usnadní sázení textu do více slopců definované šířky podobně jako v novinové sazbě.
- Transformace - transformace tvaru objektu jak ve 2D tak i ve 3D prostoru pomocí CSS. Příkladem transformací je zkosení, otočení, posun nebo rotace.

8.3 JavaScript

Je jazyk, který se v dnešní době stal nedílnou součástí webu. Oproti serverovým jazykům JavaScript běží na straně klienta. Aplikace jsou tedy spouštěny v prohlížeči u uživatele. Výhodou je jeho pohodlnost a efektivnost bez zbytečného načítání stránek a prodlev. JavaScript nikdy nebyl a ani nebude Java. Je to interpretovaný jazyk, který se překládá za běhu a vykonává podle svého zdrojového kódu.

Historie

V roce 1995 Brendan Eich vytvořil jazyk Mocha, který byl později přejmenován nejprve na LiveScript a nakonec na JavaScript. Změna názvu byla pouze obchodním tahem, jelikož z Javy přebírá jen část syntaxe, ale její filozofii ne.

V roce 1996 se firma Microsoft rozhodla, že naimplementuje pro svůj prohlížeč Internet Explorer vlastní JavaScript pod názvem JScript.

Tato technologie se začala rozšiřovat a tak byla v roce 1998 standardizována jako ECMAScript. Proces byl problematický, protože firmy W3C nebo ISO nechtěly standard schválit, a tak úkon provedla evropská firma ECMA (European Computer Manufacturers Association).

Firmě Google je přisuzována zásluha za masivní rozšíření JavaScriptu. Využila technologii AJAX založenou na tomto jazyce v aplikaci Gmail. V dnešní době neustále vznikají nové frameworky, které umožňují snadnější práci s JavaScriptem a lepší zpracování webových aplikací.[4]

Struktura

JavaScript je jazyk interpretovaný, tedy překládaný za běhu a vykonávaný podle svého zdrojového kódu. Je objektově orientovaný, ale neobsahuje nic jako třída. Objekt má tedy stejnou strukturu jako slovník. Jednou z vlastností tohoto jazyka je ukládání funkce do proměnné, neboli tzv. funkcionální paradigma.

8.4 WebGL

Web Graphics Library(WebGL) je nízkoúrovňová JavaScriptová API, která je architektonicky identická s OpenGL ES 2.0. API pro akcelerované vykreslování grafiky do HTML canvasu. Umožňuje přístup z webových stránek ke grafické kartě počítače. O vývoj a správu se stará nezisková organizace Kronos.

Specifikace WebGL 1.0 byla vydána v březnu 2011. Tomu ovšem předcházela léta vývoje. Vše započalo testování možností s elementem canvas. Experimenty začal Vladimir Vukićević ve společnosti Mozilla Foundation. Mozilla a Opera již měly své implementace v roce 2007. Vývoj nové verze WebGL 2.0 založené na OpenGL ES 3.0 započal v roce 2013.

V dnešní době je 3D prostor jednou z nejzajímavějších oblastí počítačové grafiky. Před několika lety by nás nenapadlo, že by mohly moderní prohlížeče mít takové schopnosti jako dnes. Není zapotřebí se spoléhat na zásuvné moduly třetích stran. Stačí využít JavaScriptové API pro WebGL, které poskytne komplexní 2D a 3D grafiku. Většina prohlížečů již tuto technologii podporuje, a tak není problém s používáním 3D grafiky na webových stránkách a hraní 3D her prostřednictvím webového prohlížeče. Vnitřní složitost WebGL je vysoká, a proto je práce s ní náročná. Pro usnadnění se používají nadstavby, které ale nejdou do úplné hloubky, a tak nelze využít detailnějšího nastavení. Nejznámější nadstavby jsou například Three.js, PhiloGL nebo Babylon.js.

Babylon.js

Babylon.js je open source JavaScript framework, jehož prostřednictvím lze vytvářet 3D grafiku v prohlížečích za pomoci WebGL a HTML5. Ve WebGL by vykreslení jednoduchého 3D objektu bylo napsáno v rozsáhlém množství kódu, ale Babylon.js dokáže vytvořit takovýto objekt s minimálním množstvím kódu a velmi nízkou úrovní složitosti. Ve svém prohlížeči lze vytvářet od plně interaktivní 3D her až po animovaná loga.

9 Implementace 3D editoru

Tento projekt je zaměřen na implementaci webové aplikace 3D editor, za použití technologií HTML5, CSS3 a JavaScript. Hlavní částí projektu je JavaScript a jeho podpora WebGL technologie. Jak již bylo zmíněno, tak pro práci s 3D objekty a celkově s celou scénou je použit framework BabylonJS. Z HTML5 je použit především WebSocket, který umožňuje komunikaci se serverem. Moderní design byl vytvořen za pomoci souboru nástrojů Twitter Bootstrap. Následně budou v textu popsány zajímavé nebo problematické části a jejich řešení.

9.1 Menu

Menu, jako už pojem nasvědčuje, slouží k ovládání celého editoru. S tím lze manipulovat za pomoci kurzoru myši, ale také klávesových zkratek, které zjednodušují a hlavně zrychlují práci s editorem. Odchytávání stisku kláves je nastaveno pro celý dokument. Při stisku funkce zkontroluje kód klávesy, a pokud je pro něj definována funkce, tak se provede. Na nasledující ukázce je vyobrazeno nastavení funkce pro stisk klávesy.

```
$(document).keydown(function (e) {
    //button R
    if (e.which == 114) {
        $("input[name=transform][value=Rotate]").prop('checked', true);
    }
    //button S
    if (e.which == 115) {
        $("input[name=transform][value=Scale]").prop('checked', true);
    }
    //button T
    if (e.which == 116) {
        $("input[name=transform][value=Translate]").prop('checked', true);
    }
    ...
```

Výpis 1: Ukázka klávesových zkratek

Klávesami lze ovládat tyto vlastnosti:

- přepínání mezi transformacemi (rotace - R, posunutí - T, změna rozměru - S)
- zapínání/vypínání editačního módu (Tab)
- přepínání typu transformovaného objektu (vrchol - V, plocha - F, celé těleso - M)
- přepínání ovládání na další kameru v režimu multiview (C)
- zobrazení nápovědy (N)
- zapnutí/vypnutí zobrazení síťové reprezentace tělesa

Menu by mělo být dostatečně intuitivní, aby se s ním dalo jednoduše pracovat. Některé možnosti nastavení se zobrazí až v momentě, kdy jsou potřeba. Například když nemáme vybraný objekt, který bychom chtěli transformovat, je zbytečné, aby byly zobrazeny přepínače pro nastavení typu transformované části. Některá tlačítka vyvolávají vyskakovací okna, čímž je ušetřen prostor a zpřehlednění práce s editorem.

9.2 Scéna

Scéna je podstatnou částí pro zobrazení celého prostoru pro modelování. Všechna tělesa, kamery i světla jsou registrovány ke scéně, ve které mají být zobrazeny. Pro vytvoření scény musí být provedeno pár podstatných kroků, které budou znázorněny následujícím kódem:

```
// Get the canvas element from our HTML above
  var canvas = document.querySelector("#renderCanvas");

  // Load the BABYLON 3D engine
  var engine = new BABYLON.Engine(canvas, true);

var createScene = function () {

    // Now create a basic Babylon Scene object
    var scene = new BABYLON.Scene(engine);

    // Change the scene background color to green.
    scene.clearColor = new BABYLON.Color3(0, 1, 0);

    // This creates and positions a free camera
    var camera = new BABYLON.FreeCamera("camera1", new BABYLON.Vector3(0, 5, -10),
        scene);

    // This targets the camera to scene origin
    camera.setTarget(BABYLON.Vector3.Zero());

    // This attaches the camera to the canvas
    camera.attachControl(canvas, false);

    // Leave this function
    return scene;

  };

// Now, call the createScene function that you just finished creating
  var scene = createScene();

// Register a render loop to repeatedly render the scene
  engine.runRenderLoop(function () {
    scene.render();
  });

// Watch for browser/canvas resize events
```

```
window.addEventListener("resize", function () {
  engine.resize();
});
```

Výpis 2: Vytvoření scény[10]

Nejdříve pomocí JavaScriptu získáme canvas a načteme z knihovny Babylonu engin pro 3D grafiku. Funkce na vytvoření scény nemusí obsahovat nic víc než samotné vytvoření instance scény, ale je dobré si nadefinovat alespoň jednu kameru a třeba i osvětlení do scény. Po vytvoření scény registrujeme její rendrování do rendrovacího cyklu našeho enginu. Dále je vhodné nastavit změnu rozměru při změně velikosti okna.

9.2.1 Multiview

Babylon podporuje multiview, ale s ovládáním kamer v tomto zobrazení je problém. Jelikož ovládání kamer se registruje k nějakému HTML tagu, je ovládání nastaveno nad celým canvasem. Tak by se dali ovládat pouze všechny kamery najednou, což není moc vhodný způsob. Řešením je přepínání ovládání kamer, kdy je aktivní ovládání pouze pro jednu kameru. Při přepnutí na jinou se deaktivují všechny kamery a zapne se ovládání pro novou, jak je znázorněno v následujícím kódu.

```
function changeCameraControl() {
    var cam = $('input[name=camera_opt]:checked').val();
    detacheAllCam();
    switch (cam) {
        case "main":
            camera.attachControl(canvas, false);
            break;
        case "x":
            camera1.attachControl(canvas, false);
            break;
        case "y":
            camera2.attachControl(canvas, false);
            break;
        case "z":
            camera3.attachControl(canvas, false);
            break;
    }
}
```

Výpis 3: Změna ovládání kamery

9.2.2 Osvětlení

Jak již bylo zmíněno, tak je vhodné do scény přidat nějaké osvětlení. Pro projekt bylo zvoleno vytváření bodových světel. U těch lze nastavovat jejich pozici a intenzitu svícení. V editoru je možné vytvářet větší množství bodových světel a lze je i dodatečně editovat. Bohužel to nejsou viditelné prvky, a tak jsou do scény přidány žluté kužely symbolizující

zdroj. Jsou umístěny přesně na stejných místech, jako jsou bodová osvětlení. Změna jejich pozice lze provádět stejným způsobem jako u posouvání těles ve scéně, stačí vybrat kužel symbolizující konkrétní zdroj.

9.3 Tělesa

Tělesa jsou objekty, které jsou zobrazovány ve scéně. Program obsahuje tři základní tvary a tím jsou krychle, koule a plocha. Při vytváření je zapotřebí zadat jejich název, tvar a souřadnice umístění v prostoru, viz ukázka níže.

```
var box = BABYLON.Mesh.CreateBox(newName, 2.0, scene);
box.material = newMaterial();
box.position = new BABYLON.Vector3(newX, newY, newZ);
```

Výpis 4: Vytvoření Krychle

V ukázce je funkce pro vytvoření krychle se jménem *newName* o velikosti *2.0* a vložena do scény. Poté je tělesu vytvořen nový materiál a nastavena pozice. Nový objekt se ihned vloží do scény a lze ho pro editaci vybrat dvojklikem. V takovém případě je u objektu automaticky zapnuto zobrazení jeho síťové reprezentace a celá scéna je přepnuta do editačního módu. Ten způsobí vypnutí kamer a umožní lepší práci s objektem. Také se zobrazí transformační scéna, se kterou lze ovládat změny a možnost výběru, co se bude transformovat, zda půjde o celý objekt, vrchol nebo plochu. Samozřejmě nesmí chybět možnost výběru typu transformace. Při přepínání mezi plochami, vrcholy a objektem se v záložce pro objekt zobrazují seznamy s vrcholy nebo plochami, které lze upravovat. Extruzi vrcholu (plochy), která je popsána v kapitole 9.5, lze vyvolat stiskem na tlačítko "E". Kliknutím na vrchol (plochu) se ve scéně zobrazí označení části, kterou chceme editovat. Ovládání transformací je popsáno v kapitole 9.4. Objekty lze i mazat a měnit jejich vlastnosti materiálu a nastavení textur. Podrobnější informace k nastavování materiál a textur se nachází v kapitole 9.7.

9.4 Transformace

Podstatnou částí celého editoru jsou transformace a jeho ovládání. Podle teorie, která je sepsána výše v kapitole transformace 3, jsou i zde v projektu naimplementovány tři druhy transformací. Jejich směr určují osy souřadné soustavy. Jsou to tyto transformace:

- posunutí - posouvá objekty ve scéně.
- rotace - objekty jsou otáčeny kolem jejich středového bodu.
- změna měřítka - mění měřítko(roztahuje/zužuje)

Transformace objektů jsou implementovány používanou knihovnou Babylon.js, ale ovlivnění jednotlivých vrcholů nebo ploch již samotná knihovna neumožňuje. Změna objektů neovlivňuje jednotlivé vrcholy, ale vytváří se transformační matice, která při každém renderování změní pozici vrcholů. Vrcholy si zachovávají lokální souřadnice, což je poloha

bodu od středu objektu. U vrcholů lze ovlivnit jen jejich pozici, a tak funkce pro tuto změnu používá pouze posunutí. Pro zvýraznění transformovaného vrcholu je použit nový objekt červené barvy a tvaru koule. Funkce prochází všechny vrcholy objektu, a když se dostane na vrchol, který má být transformován, změní jeho polohu ve směru, který byl funkci předán jako parametr. Poté jsou data do objektu opět uložena a je spuštěno renderování scény, které zobrazí změny na scéně. Provedení transformací u ploch je podobné jako u bodů, jen má každá plocha o dva body více. Pro zvýraznění transformované plochy je vytvořen nový objekt červené barvy a tvaru trojúhelníku. Ten je použitý, protože každý objekt je tvořen množinou trojúhelníků, jak bylo zmíněno v kapitole o reprezentaci těles 2. Nejprve se provede změna nového objektu a poté nové pozice vrcholů jsou uloženy na místa těch, které měly být změněny.

9.4.1 Ovládání transformací

Pro transformace je vytvořena zvláštní scéna, která vlastní pouze objekty znázorňující osy souřadné soustavy. Aby bylo možné transformace provádět, musí být k prvku html, ve kterém je scéna zobrazována, přiřazeno naslouchání pro stisknutí tlačítka myši, následný pohyb s ní a uvolnění tlačítka. Jak již bylo zmíněno v kapitole o scéně 9.2, tak prvkem zobrazujícím scénu je tzv. *canvas*. Nastavení naslouchání nad *canvasem* a popis prováděná funkce je znázorněna na výpisu kódu 5.

```
var onPointerDownClick = function (evt) {
    if (evt.button !== 0) {
        return;
    }
    var pickInfo = sceneAxis.pick(sceneAxis.pointerX, sceneAxis.pointerY, sceneAxis.pointerZ,
          function (mesh) { return mesh; });
    if (pickInfo.hit) {
        if (pickInfo.pickedMesh.name == "arrowX" || pickInfo.pickedMesh.name == "arrowY" ||
              pickInfo.pickedMesh.name == "arrowZ") {
            canvasAxis.onmousemove = onPointerMove; onPointerMove, false);
            transportMesh = pickInfo.pickedMesh;
            startingPoint = pickInfo.pickedPoint;
        }
    }
}
...
canvasAxis.onmousedown = onPointerDownClick;
...
```

Výpis 5: Přiřazení naslouchání a spuštěná funkce stisku tlačítka myši

Z kódu vyčteme, že funkce nejprve kontroluje, zda stisknuté tlačítko na myši bylo to levé. Následně zjišťuje informace, zda při kliknutí byl kurzor umístěn na nějakém objektu ve scéně. Pokud ano, pokračuje v kontrole. Jelikož transformace ovlivňuje objekt vzhledem k nějaké souřadné, tak zjišťujeme, zda byla některá z nich vybrána. V takovém případě zaregistrujeme funkci pro pohyb kurzoru ve scéně a uložíme si informace o tom, která osa byla vybrána společně s počátečním bodem pohybu. Funkce pro pohyb na scéně je

zaregistrována až po stisku tlačítka, protože jinak by byla volána při každém pohybu scénou i bez vybraného směru transformace. Když je tedy aktivní, tak při každém pohybu transformuje vybraný objekt na hlavní scéně. Zavolá funkci pro transformaci, které předá parametry podle toho, jaká osa byla vybrána. Velikost změny je výsledkem rozdílu počátečního bodu a bodu, na kterém se kurzor momentálně nachází. Aby se zachovala plynulost, tak počáteční bod transformace se nastaví na aktuální pozici kurzoru. Funkce pro pohyb kurzoru je znázorněna níže.

```
var onPointerMove = function (evt) {
    if (! startingPoint ) {
        return;
    }

    var current = sceneAxis.pick(sceneAxis.pointerX, sceneAxis.pointerY, sceneAxis.pointerZ).
        pickedPoint;

    if (current == null) {
        return;
    }
    else {
        switch (transportMesh.name) {
            case "arrowX":
                setTransform($('input[name=transform]:checked', '#tranForm').val(), "x", (current.x -
                    startingPoint.x)*3);
                break;
            case "arrowY":
                setTransform($('input[name=transform]:checked', '#tranForm').val(), "y", (current.y -
                    startingPoint.y)*3);
                break;
            case "arrowZ":
                setTransform($('input[name=transform]:checked', '#tranForm').val(), "z", (current.z -
                    startingPoint.z)*3);
                break;
        }
        startingPoint = current;
    }
}
```

Výpis 6: Funkce volaná při pohybu kurzoru

9.5 Extruze

Programy pro modelování a 3D grafiku (např. Blender) mohou u objektů vytvářet nové plochy nebo vrcholy pouhým vytlačením původní plochy nebo vrcholu. Tato metoda je implementována v obou variantách. U vrcholu jde o jednoduchou funkci, která z počátečního bodu vytvoří nový vrchol ve směru zvolené osy, který vloží do pole vrcholu tělesa a společně s původním vrcholem je vytvořena plocha, aby byla vidět spojitost mezi těmito body. Následně je volána funkce pro uložení nové struktury objektu. Ukázka kódu vytlačování vrcholu je níže.

```
function extrudePoint(pX, pY, pZ) {
    var positP = undefined;
    var oldP = undefined;
    var face = [];
    var f = [];
    for (var i = 0; i < vertices.length; i++) {
        if (pX == vertices[i].x && pY == vertices[i].y && pZ == vertices[i].z) {
            f.push(i);
            f.push(new BABYLON.Vector3(1, 1, 1));
            face.push(f);
        }
    }
    f = [];
    switch ($('input[name=axis]:checked', '#axisForm').val()) {
        case "x":
            var p = new BABYLON.Vector3(pX + 1, pY, pZ);
            vertices.push(p)
            f.push(vertices.length - 1);
            f.push(new BABYLON.Vector3(1, 1, 1));
            face.push(f);
            break;
        case "y":
         var p = new BABYLON.Vector3(pX, pY + 1, pZ);
          ...
         break;
        case "z":
         var p = new BABYLON.Vector3(pX, pY, pZ + 1);
          ...
         break;
    }
    map.push(face);
    setMesh();
    setMap();
}
```

Výpis 7: Vytlačení vrcholu

Vytlačování ploch je trochu složitější, jelikož je zapotřebí vytvořit nové plochy, které propojují sousední plochy té původní s novou. Podle normály původní plochy je určen směr vytlačení a zbývá vyřešit vytváření okolních ploch. Jsou známy vrcholy původní a nové, takže je možné domyslet nové plochy. Při pohledu na viditelnou stranu trojúhelníku musíme použít vrcholy proti směru hodinových ručiček (pravidlo pravé ruky). Je to proto, aby při výpočtu normálového vektoru plochy byla zobrazena správná strana stěny. Vždy by mělo být vytvořeno 6 nových stěn + jedna vytlačená.

9.6 Náhrávání ze souboru .obj

Pro nahrávání objektů ze souboru je podstatné si načíst soubor, celý jeho obsah vložit do proměnné a číst po řádcích. Celý soubor je strukturován tak, že jednotlivé objekty jsou

skládány postupně. Objet začíná názvem, dále jsou vypsány všechny vrcholy a jejich pozice a nakonec jsou definovány jednotlivé plochy. Také se v souboru mohou objevit normály a hodnoty pro mapování vrcholů. Funkce tedy projde celý soubor a postupně ukládá hodnoty do polí pro vrcholy, normály, uv hodnoty a plochy. Po vytvoření nového objektu ho naplníme novými daty, které jsme si načetli ze souboru. V případě, že nejsou uvedeny normály, jsou vypočítány funkcí. Ukázka její části je níže.

```
function computeNormal(positions, indices) {
    var normals = [];
    var positionVectors = [];
    var facesOfVertices = [];
    var index;
    for (index = 0; index < positions.length; index += 3) {
        var vector3 = new BABYLON.Vector3(positions[index], positions[index + 1], positions[index
            + 2]);
        positionVectors.push(vector3);
        facesOfVertices.push([]);
    }
    var facesNormals = [];
    for (index = 0; index < indices.length / 3; index++) {
        var i1 = indices[index * 3];
        var i2 = indices[index * 3 + 1];
        var i3 = indices[index * 3 + 2];
        var p1 = positionVectors[i1];
        var p2 = positionVectors[i2];
        var p3 = positionVectors[i3];
        var p1p2 = p1.subtract(p2);
        var p3p2 = p3.subtract(p2);
        facesNormals[index] = BABYLON.Vector3.Normalize(BABYLON.Vector3.Cross(p1p2, p3p2)
            );
      .
      .
      .
```

Výpis 8: Funkce pro výpočet normál

Funkce začíná tím, že si nejdříve vytvoří pole vrcholů a poté postupně prochází každou plochu. Z vrcholů, které do plochy patří, vypočítává směrový vektor plochy. Výpočet tohoto vektoru je následující. Nejprve si spočítáme směrové vektory hran plochy(stačí dvě). Dále uděláme vektorový součin hran a výsledkem je směrový vektor, který ještě musí být normalizován. Poté prochází postupně vrcholy a ke každému přidá do pole normál odpovídající směrový vektor. Nakonec funkce vrací pole normál. Také je možné, že nebyly uvedeny informace o mapování textur. Tento problém je také řešen funkcí pro jejich získávání. Výpočet se spouští již při ukládání jednotlivých ploch. Více o mapování textur bude v následující kapitole9.7.

9.7 Materiál a textury

U objektů nastavujeme i jejich materiál, neboli vlastnosti odrazu světla. Hlavními částmi jsou ambientní, spekulární a difúzní složka. Jejich popis byl už zmíněn ve Phongově osvětlovacím modelu.6.2. Tyto vlastnosti nastavujeme vektorem, který zastupuje jednotlivé složky barevného modelu RGB. Ve formuláři pro nastavení materiálu se nachází prvky HTML5, které usnadňují výběr barev. Vybraná barva se pak musí přepočítat na vektor, kterým nastavujeme barvu určité složky materiálu. V následujícím kódu bude znázorněno přepočítání jednotlivých složek na hodnotu v intervalu od 0.0 do 1.0 a nastavení složky materiálu.

```
function saveAmbient(val) {
    var r = (parseInt((val[1] + val[2]), 16) / 255);
    var g = (parseInt((val[3] + val[4]), 16) / 255);
    var b = (parseInt((val[5] + val[6]), 16) / 255);
    material.ambientColor = new BABYLON.Color3(r, g, b);
}
```

Výpis 9: Nastavení ambientní složky materiálu

Při výběru tělesa se zpětně přepočítávají hodnoty materiálů, aby byly správně nastaveny input tagy pro výběr barev. Větší problém přinášejí textury, které jsou sice na nastavení jednoduché, ale podmínkou správného zobrazení je namapování. V záložce pro nastavení textur se nachází obrázky s možnostmi textur. Ty lze nastavovat pouhým kliknutím na obrázek. Lze také vypnout nebo zapnout zobrazení síťové reprezentace objektu.

```
currentMesh.material.wireframe = true;
```

Výpis 10: Nastavení zobrazení síťové reprezentace tělesa

Mapování textur objektu nastavuje pro každý vrchol dvourozměrný vektor u, v. U i v nabývá hodnot v rozmezí od 0.0 do 1.0. Pokud objekt nemá nastavené UV hodnoty mapování, tak se textura ani nezobrazí. Následující funkce počítá UV hodnoty jednotlivých ploch.

```
function computeUV(faceUV) {
    if (faceUV.length < 3)
    {
        return null;
    }
    var vert = [vertices[faceUV[0][0]], vertices[faceUV[1][0]], vertices[faceUV[2][0]]];

    var uvmapxmin = vertices[0].x;
    var uvmapzmin = vertices[0].z;
    var uvmapymin = vertices[0].y;
    var uvmapxmax = vertices[0].x;
    var uvmapzmax = vertices[0].z;
    var uvmapymax = vertices[0].y;
    vertices.forEach(function (v) {
        if (v.x < uvmapxmin) {
            uvmapxmin = v.x;
```

```
    }
    else if (v.x > uvmapxmax) {
        uvmapxmax = v.x;
    }
    if (v.z < uvmapzmin) {
        uvmapzmin = v.z;
    }
    else if (v.z > uvmapzmax) {
        uvmapzmax = v.z;
    }
    if (v.y < uvmapymin) {
        uvmapymin = v.y;
    }
    else if (v.y > uvmapymax) {
        uvmapymax = v.y;
    }
});
var orient = "";
var vertminx = vert[0].x;
var vertminy = vert[0].y;
var vertminz = vert[0].z;
var vertmaxx = vert[0].x;
var vertmaxy = vert[0].y;
var vertmaxz = vert[0].z;
vert.forEach(function (v) {
    if (v.x < vertminx) {
        vertminx = v.x;
    }
    else if (v.x > vertmaxx) {
        vertmaxx = v.x;
    }
    if (v.z < vertminz) {
        vertminz = v.z;
    }
    else if (v.z > vertmaxz) {
        vertmaxz = v.z;
    }
    if (v.y < vertminy) {
        vertminy = v.y;
    }
    else if (v.y > vertmaxy) {
        vertmaxy = v.y;
    }
});

var distx = Math.abs(vertmaxx - vertminx);
var disty = Math.abs(vertmaxy - vertminy);
var distz = Math.abs(vertmaxz - vertminz);

if (distx <= distz && distx <= disty) {
    orient = "x";
}
else if (disty <= distz && disty <= distx) {
    orient = "y";
```

```
        }
        else if (distz <= distx && distz <= disty) {
            orient = "z";
        }

        var uvs = [];
        vert.forEach(function (v) {
            var uv = {u:0, v:0};
            switch(orient){
                case "x":
                    uv.u = (v.y - uvmapymin) / (uvmapymax - uvmapymin);
                    uv.v = (v.z - uvmapzmin) / (uvmapzmax - uvmapzmin);
                    uvs.push(uv);
                    break;
                case "y":
                    uv.u = (v.x - uvmapxmin) / (uvmapxmax - uvmapxmin);
                    uv.v = (v.z - uvmapzmin) / (uvmapzmax - uvmapzmin);
                    uvs.push(uv);
                    break;
                case "z":
                    uv.u = (v.x - uvmapxmin) / (uvmapxmax - uvmapxmin);
                    uv.v = (v.y - uvmapymin) / (uvmapymax - uvmapymin);
                    uvs.push(uv);
                    break;
            }
        });

        return uvs;
    }
```

Výpis 11: Funkce pro výpočet uv hodnot plochy

Funkce si nejprve zjistí nejmenší hodnoty souřadnic jednotlivých os ze všech bodů objektu. To samé provede pro body plochy, aby si zjistila, jakým směrem je plocha orientována. Nejmenší absolutní hodnota rozdílu maximální s minimální hodnotou souřadnice určuje, ke které ose je plocha více kolmá. U výpočtu hodnot pro mapování potřebujeme pouze hodnoty dvou souřadných os, jelikož textura je dvourozměrná, a proto je předchozím výpočtem jedna souřadná osa vyloučena. Funkce tedy pokračuje výpočtem UV hodnot pro každý vrchol plochy. Funkce vrací pole těchto hodnot pro danou plochu.

9.8 Export objektů

Výsledek své práce v editoru si bude chtít uživatel nejspíše uložit, a tak jsou tu funkce vytvořené pro export objektů ze scény do obj. souboru. Princip je jednoduchý, jelikož jde jen o postupném ukládání všech objektů ze scény. Proto funkce prochází cyklicky všechny objekty a u každého objektu do souboru zapisuje informace o něm samotném. Začíná jménem, pokračuje výpisem vrcholů přes normály a končí plochami. Výsledkem celého exportu je ZIP soubor, který obsahuje .obj soubor a použité textury. Takže se nejprve vytvoří nový archiv pomocí knihovny JSZIP, poté se do něj vloží .obj soubor s celým řetězcem obsahující data objektů a nakonec je vytvořena složka s texturami a pro každé

těleso, které má ve scéně nějakou texturu, uložíme obrázek této textury do složky. Funkce vrací blob soubor, který se buď uloží na disk, nebo zašle prostřednictvím socketu.

```
function createZip() {
    var zip = new JSZip();
    var name = $("#fileName").val();
    zip.file(name + ".obj", createFile());
    zip.folder("textury");
    meshes.forEach(function (m) {
        if (m.material.diffuseTexture != null) {
            var url = m.material.diffuseTexture.url;
            var n = url.split("/");
            var textFile = convertImgToBase64(url);
            console.info(textFile);
            zip.file("textury/" + n[n.length - 1], textFile, { base64: true });
        }
    });
    var content = zip.generate({ type: "blob" });

    return content;
}
```

Výpis 12: Vytvoření ZIP archivu

9.8.1 Lokální disk

Ukládání nejprve zahájí vytvoření ZIP archivu. Jakmile je hotov a předán, tak je vytvořen HTML element typu odkaz a jsou mu přiřazeny vlastnosti pro stažení (název souboru a odkaz na vytvoření souboru z URL). Na závěr funkce je pomocí JavaScriptu kliknuto na odkaz.

```
function saveFileToLocal() {
    var fileAsBlob = createZip();
    var name = $("#fileName").val();
    var downloadLink = document.createElement("a");
    downloadLink.download = name + ".zip";
    downloadLink.innerHTML = "Download␣File";
    if (window.webkitURL != null) {
        downloadLink.href = window.webkitURL.createObjectURL(fileAsBlob);
    }
    else {
        downloadLink.href = window.URL.createObjectURL(fileAsBlob);
        downloadLink.onclick = destroyClickedElement;
        downloadLink.style.display = "none";
        document.body.appendChild(downloadLink);
    }

    downloadLink.click();
    $("#saveLocalFile").hide();
}
```

Výpis 13: Uložení souboru na disk

9.8.2 WebSocket

Pro ukládání prostřednictvím WebSicket začíná stejně jako při ukládání na lokální disk. Nejprve se vytvoří ZIP archiv a dále je vytvořeno spojení se serverem. Pro WebSocket jsou nastaveny funkce na to, jak má aplikace reagovat v případě otevření spojení a vyvolání chyby. Samozřejmě při otevření spojení zasíláme soubor pomocí funkce *send*.

```
function saveFileWithWebSocket() {
    var fileAsBlob = createZip();
    var name = $("#fileName").val();
    var socket = $("#webSocketServer").val();
    var ws = new WebSocket(socket);

    ws.onopen = function () {
        ws.send(fileAsBlob); //send a message to server once connection is opened.
    };
    ws.onerror = function (error) {
        console.log('Error Logged: ' + error); //log errors
    };
    ws.onmessage = function (e) {
        console.log('Received From Server: ' + e.data); //log the received message
    };

    $("#saveWebSocketFile").hide();
}
```

Výpis 14: Uložení souboru pomocí WebSocket

10 Závěr

Výsledkem práce je webový 3D editor, který ukazuje možnosti WebGL technologie ve webových prohlížečích za použití JavaScriptu. Editor pracuje s objekty reprezentovány sítí trojúhelníků. Součástí implementace je vytváření těles ve scéně a jejich transformace, přidávání bodových světel, zobrazení scény z více pohledů, nastavování materiálů a textur, načítání objektů ze souboru a ukládání objektů scény do souboru a ukládání na lokální disk nebo odesílání prostřednictvím socketů. Při řešení všech těchto částí jsem nabyl spoustu nových znalostí, které jsem aplikoval při implementaci.

Editor lze využít k modelování vlastních těles nebo těles vyskytujících se v reálném světě. Bohužel jsou funkce editoru omezené oproti jiným sofistikovaným programům zaměřeným na modelování. Jeho využití bych viděl spíše k seznámení začínajících grafiků s principy modelování a práce s 3D grafikou.

Do budoucna bych se chtěl této aplikaci nadále věnovat a vytvořit produkt, který by se dal využít například v bytovém designu. Rozšířit editor o knihovnu konkrétních objektů reálného světa (např. stůl, židle, police, atd.), přidat více textur.

11 Reference

[1] Mozila. *Slovník terminologie* [online]. [cit. 2015-03-24]. Dostupné z: https://support.mozilla.org/cs/kb/Slovn

[2] BROWN, Tiffany B, Kerry BUTTERS a Sandeep PANDA.*HTML5 okamžitě: výukový kurz webového vývojáře.* 1. vyd. Brno: Computer Press, 2014, 256 s. ISBN 978-80-251-4296-7.

[3] LAZARIS, Louis, Kerry BUTTERS a Sandeep PANDA. *CSS Okamžitě: výukový kurz webového vývojáře.* 1. vyd. Brno: Computer Press, 2014, 168 s. ISBN 978-80-251-4176-2.

[4] PEHLIVANIAN, Ara, Don NGUYEN a Sandeep PANDA. *JavaScript okamžitě: výukový kurz webového vývojáře.* 1. vyd. Brno: Computer Press, 2014, 160 s. ISBN 978-80-251-4163-2.

[5] ŽÁRA, Jiří. *Moderní počítačová grafika.* 2., přeprac. a rozš. vyd. Praha: Computer Press, 2004, 609 s., 16 s. barev. obr. příl. ISBN 8025104540.

[6] *Computer Graphics and Visualisation.* [online]. [cit. 2015-04-19]. Dostupné z: http://www.pling.org.uk/cs/cgv.html

[7] SLABÝ, Antonín. *Počítačová grafika.* Vyd. 1. Hradec Králové: Gaudeamus, 2012, 1 CD-ROM. ISBN 978-80-7435-208-9.

[8] LINKEOVÁ, Ivana. *Promítací metody.* [online]. 2014 [cit. 2015-05-02]. Dostupné z: http://www.linkeova.cz/jsMath/ApGeom/proj.pdf

[9] REDDY, Martin. *Object files.* [online]. [cit. 2015-05-02]. Dostupné z: http://www.martinreddy.net/gfx/3d/OBJ.spec

[10] *Basic scene* [online]. [cit. 2015-05-05]. Dostupné z: https://github.com/BabylonJS/Babylon.js/wiki/01—Basic-scene

[11] *OpenGL Projection Matrix* [online]. [cit. 2015-05-05]. Dostupné z: http://www.songho.ca/opengl/gl_projectionmatrix.html

[12] *Phong Illumination*: obrázek. [online]. [cit. 2015-04-19]. Dostupné z: http://www.pling.org.uk/cs/cgvimg/specularlighting.png

[13] *Nonmanifold*: obrázek. [online]. [cit. 2015-04-28]. Dostupné z: http://www.meshrepair.org/meshrep_file/botsch.jpg

[14] *Středové promítání*: obrázek. [online]. [cit. 2015-05-02]. Dostupné z: http://www.surynkova.info/dokumenty/materials/lp/lp.jpg

[15] *Perspektiva*: obrázek. [online]. [cit. 2015-05-02]. Dostupné z: http://3dgrafikabalusek.wz.cz/images/perspektiva.png

A Obsah přílohy CD

Obsah složek:

- "3DEditor"- program se zdrojovými kódy.
- "OBJ"- .obj soubory použité k testování.
- Dokumentace - programátorská dokumentace projektu.